UNIVERSITY AT ALBANY
State University of New York

COLLEGE OF ENGINEERING AND APPLIED SCIENCES
DEPARTMENT OF COMPUTER SCIENCE
CSI213 Data Structures

Project 04 Created by Qi Wang

Click here for the project discussion recording.

Table of Contents

Part I: General Project Information

- All projects are individual projects unless it is notified otherwise.
- All projects must be submitted via Blackboard. No late projects or e-mail submissions or hard copies will be accepted.
- Two submission attempts will be allowed on Blackboard. **Only the last attempt will be graded.**
- Work will be rejected with no credit if
    - The project is late.
    - The project is not submitted properly (wrong files, not in required format, etc.). For example,
        - The submitted file can't be opened.
        - The submitted work is empty or wrong work.
        - Other issues.
    - The project is a copy or partial copy of others' work (such as work from another person or the Internet).
- **Students must turn in their original work. Any cheating violation will be reported to the college. Students can help others by sharing ideas, but not by allowing others to copy their work.**
- Documents to be submitted as a zipped file:
    - **UML class diagram(s)** – created with Violet UML or StarUML
    - **Java source file(s) with Javadoc style inline comments**
    - **Supporting files if any** (For example, files containing all testing data.)
- Students are required to submit a design, all error-free source files with Javadoc style inline comments, and supporting files. Lack of any of the required items will result in a really low credit or no credit.
- Your TA will grade, and then post the feedback and the grade on Blackboard if you have submitted it properly and on time. If you have any questions regarding the feedback or the grade, please reach out to the TA first. You may also contact the instructor for this matter.

Part II: Project grading rubric

Components	**Max points**
UML Design (See an example in part II.)	Max. 10 points
Javadoc Inline comments (See an example in part II.)	Max. 10 points
The rest of the project	Max. 40 points

All projects will be evaluated based upon the following software development activities.

Analysis:

- Does the software meet the exact specification / customer requirements?
- Does the software solve the exact problem?

Design:

- Is the design efficient?

Code:

- Are there errors?
- Are code conventions followed?
- Does the software use the minimum computer resource (computer memory and processing time)?
- Is the software reusable?
- Are comments completely written in Javadoc format?
    a. Class comments must be included in Javadoc format before a class header.
    b. Method comments must be included in Javadoc format before a method header.
    c. More inline comments must be included in either single line format or block format inside each method body.
    d. All comments must be completed in correct format such as tags, indentation etc.

Debug/Testing:

- Are there bugs in the software?

Documentation:

- Complete all documentations that are required.

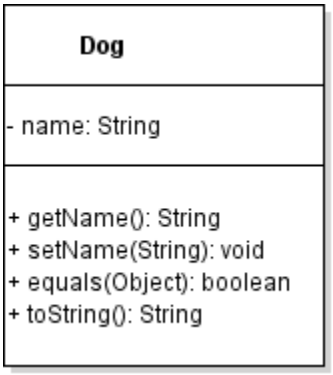
Part III: Examples on how to complete a project from start to finish

To complete a project, the following steps of a software development cycle should be followed. These steps are not pure linear but overlapped.

Analysis-design-code-test/debug-documentation.

1) Read project description to understand all specifications(**Analysis**).
2) Create a design (an algorithm for method or a UML class diagram for a class) (**Design**)
3) Create Java programs that are translations of the design. (**Code/Implementation**)
4) Test and debug, and (**test/debug)**
5) Complete all required documentation. (**Documentation**)

The following shows a sample design. The corresponding source codes with inline Javadoc comments are included on next page. How to test/debug a software is included on the following pages.



```
import java.util.Random;

/**
 * Representing a dog with a name.
 * @author Qi Wang
 * @version 1.0
 */
public class Dog{
    /**
     * The name of this dog
     */
    private String name;


    /**
     * Constructs a newly created Dog object that represents a dog with an empty name.
     */
    public Dog(){
        this("");
    }
    /**
     * Constructs a newly created Dog object wit
     * @param name The name of this dog
     */
    public Dog(String name){
        this.name = name;
    }

    /**
     * Returns the name of this dog.
     * @return The name of this dog
     */
    public String getName(){
        return this.name;
    }

    /**
     * Changes the name of this dog.
     * @param name The name of this dog
     */
    public void setName(String name){
        this.name = name;
    }

    /**
     * Returns a string representation of this dog. The returned string contains the type of
     * this dog and the name of this dog.
     * @return A string representation of this dog
     */
    public String toString(){
        return this.getClass().getSimpleName() + ": " + this.name;
    }


    /**
     * Indicates if this dog is "equal to" some other object. If the other object is a dog,
     * this dog is equal to the other dog if they have the same names. If the other object is
     * not a dog, this dog is not equal to the other object.
     * @param obj A reference to some other object
     * @return A boolean value specifying if this dog is equal to some other object
     */
    public boolean equals(Object obj){
        //The specific object isn't a dog.
        if(!(obj instanceof Dog)){
            return false;
```

TAB

TAB

TAB

TAB

open {

open {

Class comments must be written in Javadoc format before the class header. A **description** of the class, author information and version information are required.

Comments for fields are required.

Method comments must be written in Javadoc format before the method header. the first word must be a **capitalized** verb in the **third** person. Use punctuation marks properly.

A description of the method, comments on parameters if any, and comments on the return type if any are required.

A Javadoc comment for a formal parameter consists of three parts:

- parameter tag,
- a name of the formal parameter in the design ,
  (The name must be consistent in the comments and the header.)
- and a phrase explaining what this parameter specifies.

A Javadoc comment for return type consists of two parts:

- return tag,
- and a phrase explaining what this returned value specifies

More inline comments can be included in single line or block comments format in a method.

```
            }

            //The specific object is a dog.
            Dog other = (Dog)obj;
            return this.name.equalsIgnoreCase(other.name);
        }
}
```

Part IV:

A. How to test a software design?

There can be many classes in a software design.

1. **First, create a UML class diagram containing the designs of all classes and the class relationships (For example, is-a, dependency or aggregation).**
2. **Next, test each class separately.**
   Convert each class in the diagram into a Java program. When implementing each class, a driver is needed to test each method included in the class design. In the driver program,
   i. Use the constructors to create instances of the class(If a class is abstract, the members of the class will be tested in its subclasses.). For example, the following creates Dog objects.
   Create a default Dog object.
   `Dog firstDog = new Dog();`
   Create a Dog object with a specific name.
   `Die secondDog = new Dog("Sky");`

   ii. Use object references to invoke the instance methods. If an instance method is a value-returning method, call this method where the returned value can be used. For example, method getName can be called to return a copy of firstDog's name.

```
String firstDogName;
  …
firstDogName = firstDog.getName();
```

   You may print the value stored in firstDogName to verify.

   iii. If a method is a void method, invoke the method that simply performs a task. Use other method to verify the method had performed the task properly. For example, setName is a void method and changes the name of this dog. After this statement, the secondDog's name is changed to "Blue".
   `secondDog.setName("Blue");`

   getName can be used to verify that setName had performed the task.

   iv. Repeat until all methods are tested.

- **And then, test the entire design by creating a driver program and a helper class of the driver program.**
  i. Create a helper class. In the helper class, minimum three static methods should be included.

```
public class Helper{

    //method 1
    public static void start(){
        This void method is decomposed.
        It creates an empty list.
        It calls the create method to add a list of objects to the list.
        And then, it calls the display method to display the list of objects.
    }

    //method 2
    public static returnTypeOrVoid create(parameters if any) {
```

```
            This method creates a list of objects using data stored in text files.
        }

        //method 3
        public static returnTypeOrVoid display(parameters if any) {
            This method displays a list of objects.
        }
    }
```

ii. Create a driver program. In *main* of the driver program, call method *start* to start the entire testing process.

```
public class Driver{
    public static void main(String[] args){
        Helper.start();
    }
}
```

Notice that the driver and its helper class are for testing purpose only. They should not be included in the design diagram.

B: Project description

Project 4 An address book

Design **an ADT Address book** that can be used to maintain contact information of friends and families. Each contact contains a name (first name and last name), an **address** (street, city, state and zip code), and a telephone number. For example, the following shows the contact information for James Butt and Josephine Darakjy.

first_name	last_name	street	city	state	zipcode	phone
James	Butt	6649 N Blue Gum St	New Orleans	LA	70116	504-621-8927
Josephine	Darakjy	4 B Blue Ridge Blvd	Brighton	MI	48116	810-292-9388

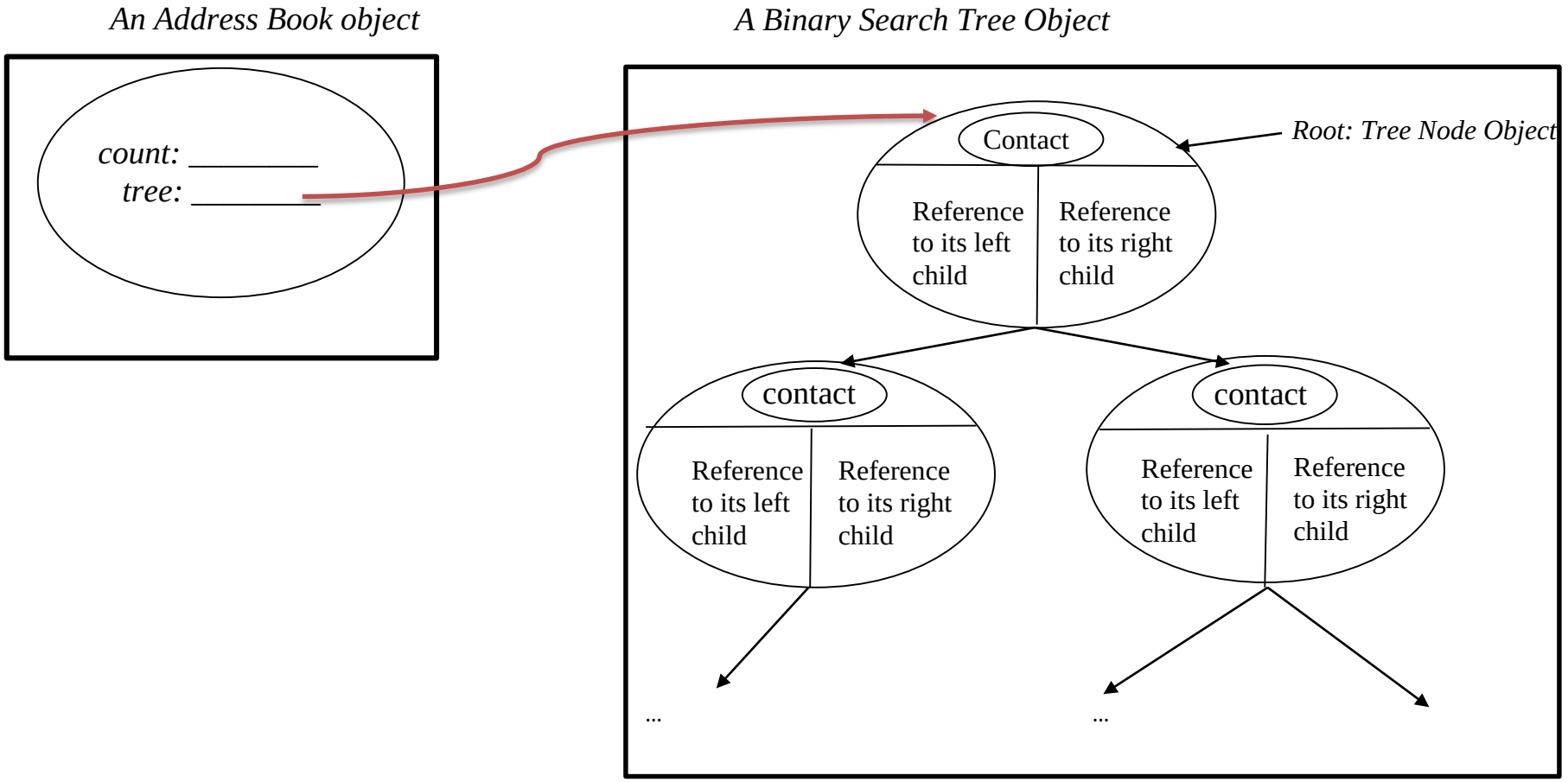
In an address book, contacts are stored in a binary search tree. Users can insert, delete, and search, get contact information or check information regarding the address book. Assume that all contact names are unique. When searching a contact in an address book, the unique name should be used as the search key.

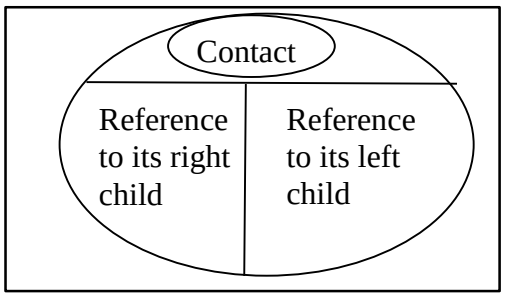
Specification (ADT Address Book)

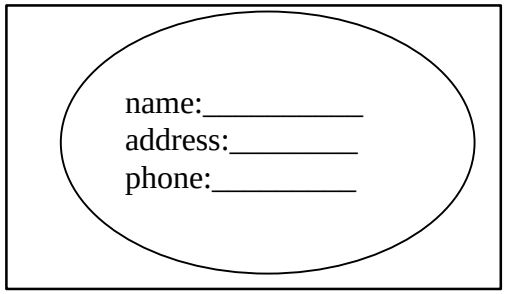
Specify operations to

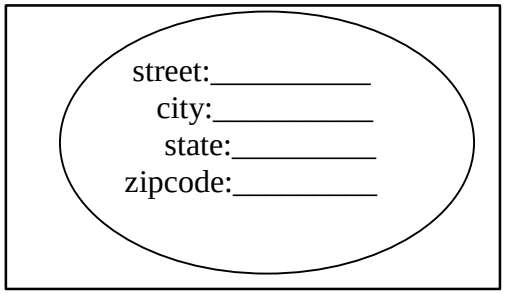
- create an empty address book,
- insert a new contact into this address book,
- delete a contact from this address book,
- search a contact in this address book,
- get a reference to a contact from this address book,
- check how many contacts are there in this address book,
- check to see if the address book is empty,
- and completely empty this address book.

The following shows the containment hierarchy of the objects that are involved in this design. An address book object contains a reference to a binary search tree. Each node of the binary search tree contains a reference to a contact object, a reference to its left child and a reference to its right child. The contact object contains a reference to a name, a reference to an address, and a reference to a telephone number.



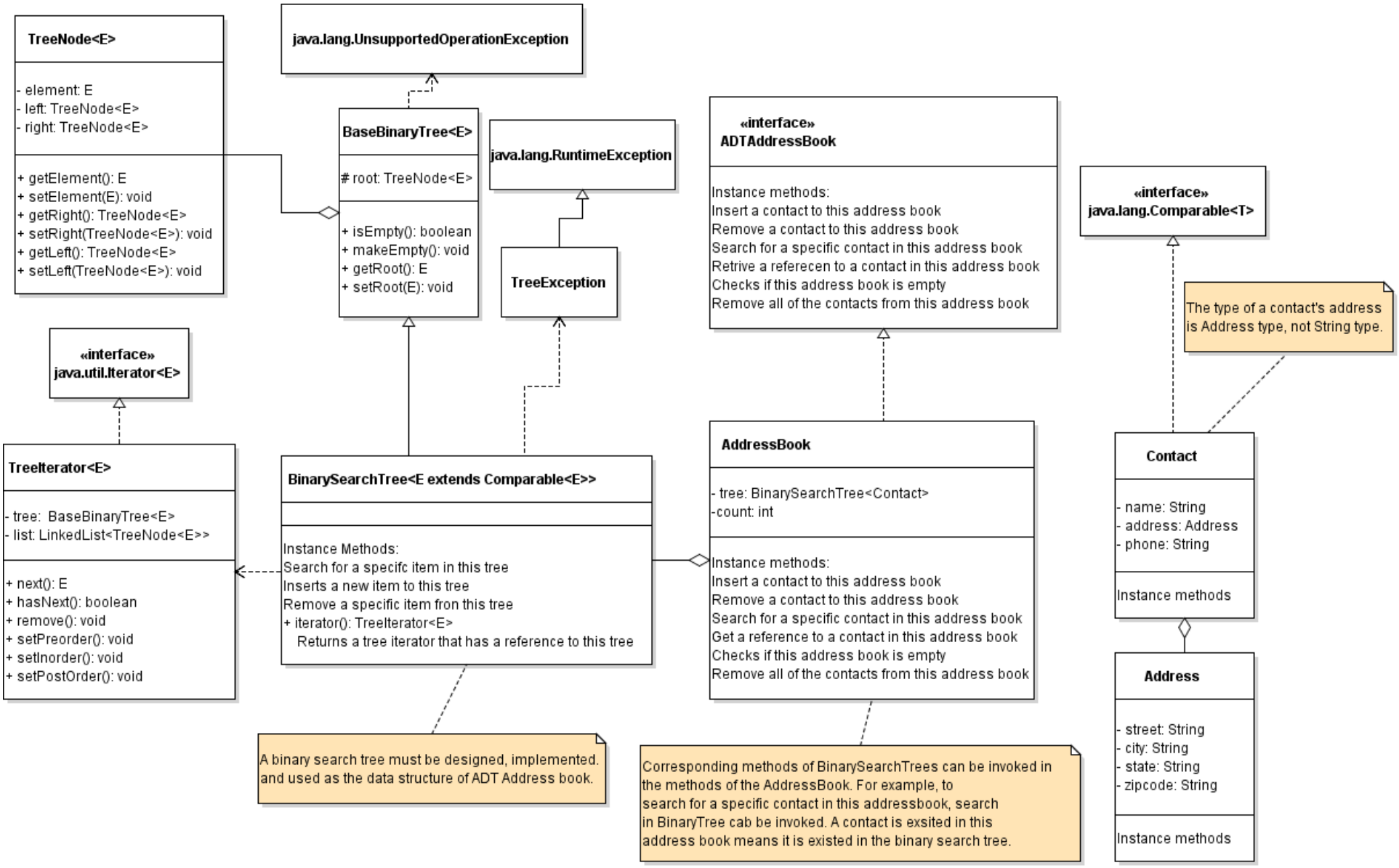






Design:

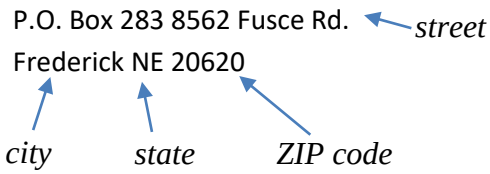
The following shows partial design of the ADT Address Book. A complete design must be submitted as part of the project submission.



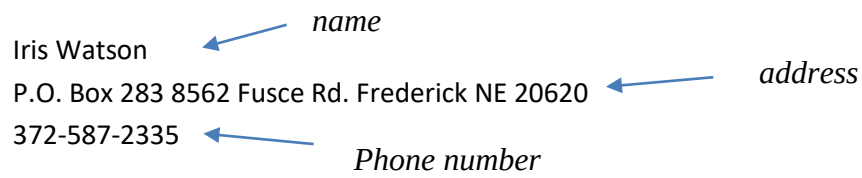
Class Address defines an address contained in a contact in an address book. Class Contact defines a contact object contained in an address book. Contacts are stored in a binary search tree in an address book. A generic ADT binary search tree must be designed and implemented so that it can be used as the data structure of the ADT Address Book. Note: It is NOT allowed to use any predefined binary search trees from Java or any other programming languages. Students are required to design and implement an ADT binary search tree as part of the project. Failing to meet the requirements may result in a very low credit or no credit.

Note: Since all the contacts are stored in the binary search tree of an address book, insertion, deletion, search, and retrieval in this address book should be conducted in the binary search tree of this address book.

- **Address:**
  Class Address represents a mailing address. An address object contains street, city, state, and zip code
  For example, the following show a full address.



- **Contact:**
  Class Contact represents a contact in an address book. A contact object contains the contact's name, address, and a phone number.



- **ADT Binary Search Tree**
  A generic class binary search tree needs to be designed. A binary search tree can be used to store a list of **comparable** objects of any types. To constrain E, a type parameter in a generic class, to be classes that implement *Comparable* interface, in *BinarySearchTree*, add *E extends Comparable<E>* into the bracket in the class header.

```
public class BinarySearchTree<E extends Comparable<E>>
```

  The constraint <E extends AClass > establishes AClass as an upper bound for E. The constraint says E may be any subclass of AClass. A contact is stored in a node of a binary search tree. A generic TreeNode class must be designed.

  Each list type, such as a binary search tree, needs an iterator with which a traversal in preorder, inorder or postorder can be done, Therefore, the method iterator needs to be overridden in the binary search tree class. This method should return an instance of a tree iterator with a reference to this binary search tree.

  An empty binary search tree should be instantiated. Nodes can be inserted to or removed from a binary search tree. Other operations such as search, etc. can be done in a binary search tree.

- **ADT Address Book**
  Class AddressBook represents an ADT address book. An address book is required to be implemented using data structure, a binary search tree. When using a binary search tree to store the contacts of an address book, the type parameter can be replaced with an actual type, Contact.
- Other classes needed to complete the ADT Address Book design.

Code:
A generic binary search tree must be designed. A binary search tree with element type Contact is used as the data structure in the ADT Address Book design. All source files must include Javadoc style comments.

Debug/Test:
To test the project, a driver with a helper class is needed. In the helper class, use the testing data to create an address book containing the original list of contacts, and test the entire design completely. The following contains starter source codes for the helper class.

```
public class … {
	public static void start()…{
		//Create an empty address book
		//In a method, fill the address book.
		//In a method, display the address book.
		//More…
	}
}
```

Note: The following testing data is saved into a file named contacts.txt. A tab is used as the delimiter.

first_name	last_name	street	city	state	zipcode	phone
James	Butt	6649 N Blue Gum St	New Orleans	LA	70116	504-621-8927
Josephine	Darakjy	4 B Blue Ridge Blvd	Brighton	MI	48116	810-292-9388
Art	Venere	8 W Cerritos Ave #54	Bridgeport	NJ	8014	856-636-8749
Lenna	Paprocki	639 Main St	Anchorage	AK	99501	907-385-4412
Donette	Foller	34 Center St	Hamilton	OH	45011	513-570-1893
Simona	Morasca	3 Mcauley Dr	Ashland	OH	44805	419-503-2484
Mitsue	Tollner	7 Eads St	Chicago	IL	60632	773-573-6914
Leota	Dilliard	7 W Jackson Blvd	San Jose	CA	95111	408-752-3500
Sage	Wieser	5 Boston Ave #88	Sioux Falls	SD	57105	605-414-2147
Kris	Marrier	228 Runamuck Pl #2808	Baltimore	MD	21224	410-655-8723
Minna	Amigon	2371 Jerrold Ave	Kulpsville	PA	19443	215-874-1229
Abel	Maclead	37275 St Rt 17m M	Middle Island	NY	11953	631-335-3414
Kiley	Caldarera	25 E 75th St #69	Los Angeles	CA	90034	310-498-5651
Graciela	Ruta	98 Connecticut Ave Nw	Chagrin Falls	OH	44023	440-780-8425
Cammy	Albares	56 E Morehead St	Laredo	TX	78045	956-537-6195
Mattie	Poquette	73 State Road 434 E	Phoenix	AZ	85013	602-277-4385
Meaghan	Garufi	69734 E Carrillo St	Mc Minnville	TN	37110	931-313-9635
Gladys	Rim	322 New Horizon Blvd	Milwaukee	WI	53207	414-661-9598
Yuki	Whobrey	1 State Route 27	Taylor	MI	48180	313-288-7937
Fletcher	Flosi	394 Manchester Blvd	Rockford	IL	61109	815-828-2147
Bette	Nicka	6 S 33rd St	Aston	PA	19014	610-545-3615
Veronika	Inouye	6 Greenleaf Ave	San Jose	CA	95111	408-540-1785
Willard	Kolmetz	618 W Yakima Ave	Irving	TX	75062	972-303-9197

Maryann	Royster	74 S Westgate St	Albany	NY	12204	518-966-7987
Alisha	Slusarski	3273 State St	Middlesex	NJ	8846	732-658-3154
Allene	Iturbide	1 Central Ave	Stevens Point	WI	54481	715-662-6764
Chanel	Caudy	86 Nw 66th St #8673	Shawnee	KS	66218	913-388-2079
Ezekiel	Chui	2 Cedar Ave #84	Easton	MD	21601	410-669-1642
Willow	Kusko	90991 Thorburn Ave	New York	NY	10011	212-582-4976
Bernardo	Figeroa	386 9th Ave N	Conroe	TX	77301	936-336-3951
Ammie	Corrio	74874 Atlantic Ave	Columbus	OH	43215	614-801-9788
Francine	Vocelka	366 South Dr	Las Cruces	NM	88011	505-977-3911
Ernie	Stenseth	45 E Liberty St	Ridgefield Park	NJ	7660	201-709-6245
Albina	Glick	4 Ralph Ct	Dunellen	NJ	8812	732-924-7882
Alishia	Sergi	2742 Distribution Way	New York	NY	10025	212-860-1579
Solange	Shinko	426 Wolf St	Metairie	LA	70002	504-979-9175
Jose	Stockham	128 Bransten Rd	New York	NY	10011	212-675-8570
Rozella	Ostrosky	17 Morena Blvd	Camarillo	CA	93012	805-832-6163
Valentine	Gillian	775 W 17th St	San Antonio	TX	78204	210-812-9597
Kati	Rulapaugh	6980 Dorsett Rd	Abilene	KS	67410	785-463-7829
Youlanda	Schemmer	2881 Lewis Rd	Prineville	OR	97754	541-548-8197
Dyan	Oldroyd	7219 Woodfield Rd	Overland Park	KS	66204	913-413-4604
Roxane	Campain	1048 Main St	Fairbanks	AK	99708	907-231-4722
Lavera	Perin	678 3rd Ave	Miami	FL	33196	305-606-7291
Erick	Ferencz	20 S Babcock St	Fairbanks	AK	99712	907-741-1044
Fatima	Saylors	2 Lighthouse Ave	Hopkins	MN	55343	952-768-2416
Jina	Briddick	38938 Park Blvd	Boston	MA	2128	617-399-5124
Kanisha	Waycott	5 Tomahawk Dr	Los Angeles	CA	90006	323-453-2780
Emerson	Bowley	762 S Main St	Madison	WI	53711	608-336-7444
Blair	Malet	209 Decker Dr	Philadelphia	PA	19132	215-907-9111
Brock	Bolognia	4486 W O St #1	New York	NY	10003	212-402-9216
Lorrie	Nestle	39 S 7th St	Tullahoma	TN	37388	931-875-6644
Sabra	Uyetake	98839 Hawthorne Blvd #6101	Columbia	SC	29201	803-925-5213
Marjory	Mastella	71 San Mateo Ave	Wayne	PA	19087	610-814-5533
Karl	Klonowski	76 Brooks St #9	Flemington	NJ	8822	908-877-6135
Tonette	Wenner	4545 Courthouse Rd	Westbury	NY	11590	516-968-6051
Amber	Monarrez	14288 Foster Ave #4121	Jenkintown	PA	19046	215-934-8655
Shenika	Seewald	4 Otis St	Van Nuys	CA	91405	818-423-4007
Delmy	Ahle	65895 S 16th St	Providence	RI	2909	401-458-2547
Deeanna	Juhas	14302 Pennsylvania Ave	Huntingdon Valley	PA	19006	215-211-9589
Blondell	Pugh	201 Hawk Ct	Providence	RI	2904	401-960-8259
Jamal	Vanausdal	53075 Sw 152nd Ter #615	Monroe Township	NJ	8831	732-234-1546
Cecily	Hollack	59 N Groesbeck Hwy	Austin	TX	78731	512-486-3817
Carmelina	Lindall	2664 Lewis Rd	Littleton	CO	80126	303-724-7371
Maurine	Yglesias	59 Shady Ln #53	Milwaukee	WI	53214	414-748-1374
Tawna	Buvens	3305 Nabell Ave #679	New York	NY	10009	212-674-9610
Penney	Weight	18 Fountain St	Anchorage	AK	99515	907-797-9628
Elly	Morocco	7 W 32nd St	Erie	PA	16502	814-393-5571
Ilene	Eroman	2853 S Central Expy	Glen Burnie	MD	21061	410-914-9018

Vallie	Mondella	74 W College St	Boise	ID	83707	208-862-5339
Kallie	Blackwood	701 S Harrison Rd	San Francisco	CA	94104	415-315-2761
Johnetta	Abdallah	1088 Pinehurst St	Chapel Hill	NC	27514	919-225-9345
Bobbye	Rhym	30 W 80th St #1995	San Carlos	CA	94070	650-528-5783
Micaela	Rhymes	20932 Hedley St	Concord	CA	94520	925-647-3298
Tamar	Hoogland	2737 Pistorio Rd #9230	London	OH	43140	740-343-8575
Moon	Parlato	74989 Brandon St	Wellsville	NY	14895	585-866-8313
Laurel	Reitler	6 Kains Ave	Baltimore	MD	21215	410-520-4832
Delisa	Crupi	47565 W Grand Ave	Newark	NJ	7105	973-354-2040
Viva	Toelkes	4284 Dorigo Ln	Chicago	IL	60647	773-446-5569
Elza	Lipke	6794 Lake Dr E	Newark	NJ	7104	973-927-3447
Devorah	Chickering	31 Douglas Blvd #950	Clovis	NM	88101	505-975-8559
Timothy	Mulqueen	44 W 4th St	Staten Island	NY	10309	718-332-6527
Arlette	Honeywell	11279 Loytan St	Jacksonville	FL	32254	904-775-4480
Dominque	Dickerson	69 Marquette Ave	Hayward	CA	94545	510-993-3758
Lettie	Isenhower	70 W Main St	Beachwood	OH	44122	216-657-7668
Myra	Munns	461 Prospect Pl #316	Euless	TX	76040	817-914-7518
Stephaine	Barfield	47154 Whipple Ave Nw	Gardena	CA	90247	310-774-7643
Lai	Gato	37 Alabama Ave	Evanston	IL	60201	847-728-7286
Stephen	Emigh	3777 E Richmond St #900	Akron	OH	44302	330-537-5358
Tyra	Shields	3 Fort Worth Ave	Philadelphia	PA	19106	215-255-1641
Tammara	Wardrip	4800 Black Horse Pike	Burlingame	CA	94010	650-803-1936
Cory	Gibes	83649 W Belmont Ave	San Gabriel	CA	91776	626-572-1096
Danica	Bruschke	840 15th Ave	Waco	TX	76708	254-782-8569
Wilda	Giguere	1747 Calle Amanecer #2	Anchorage	AK	99501	907-870-5536
Elvera	Benimadh	99385 Charity St #840	San Jose	CA	95110	408-703-8505
Carma	Vanheusen	68556 Central Hwy	San Leandro	CA	94577	510-503-7169
Malinda	Hochard	55 Riverside Ave	Indianapolis	IN	46202	317-722-5066
Natalie	Fern	7140 University Ave	Rock Springs	WY	82901	307-704-8713
Lisha	Centini	64 5th Ave #1153	Mc Lean	VA	22102	703-235-3937